Handy Solinan

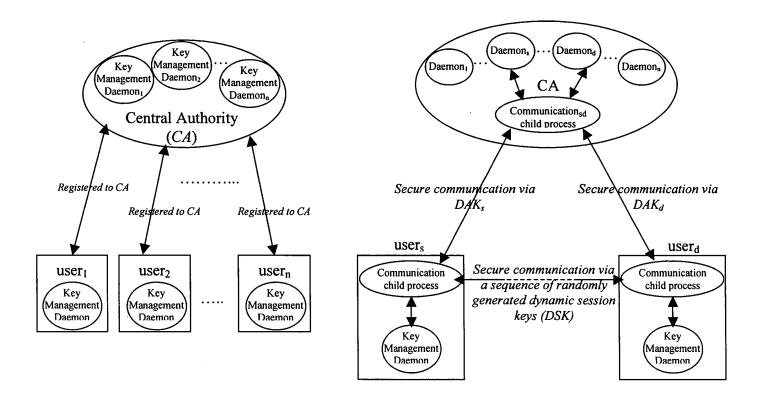


FIG. 1a

FIG. 1b

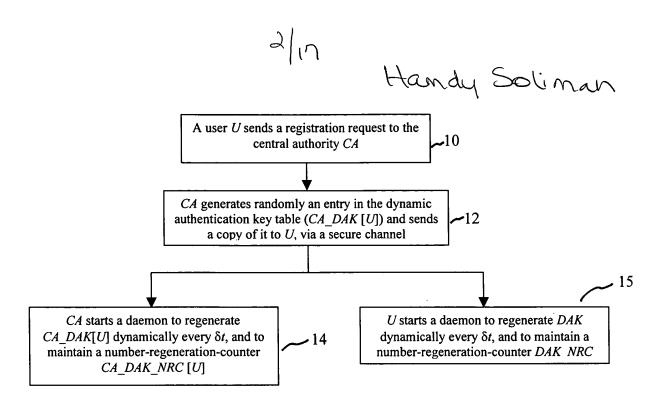


FIG. 2

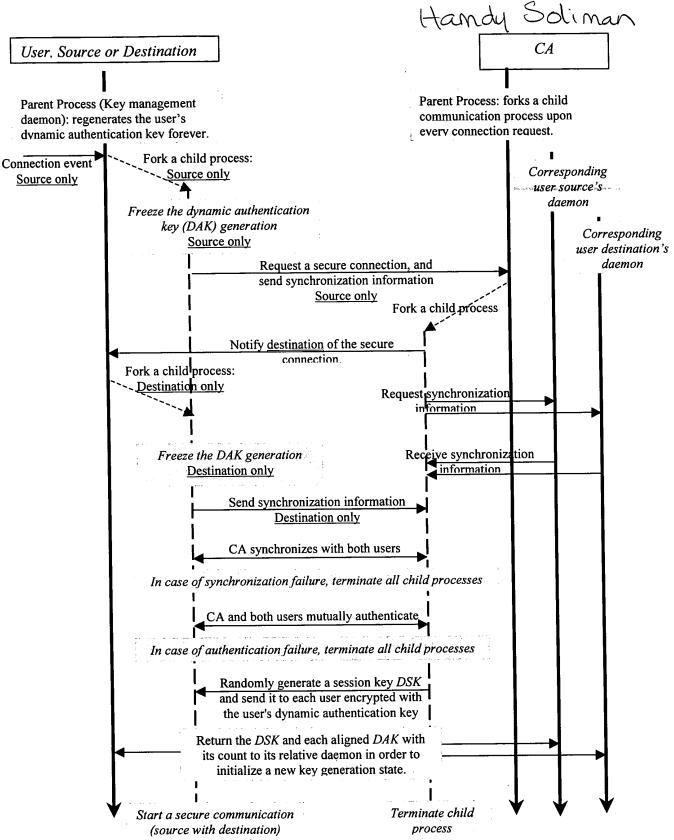


FIG. 3

4)(17

Handy Solinar CA receives a dynamic session key generation request from a user U_s to communicate with user U_d , along with its frozen $U_s_DAK_NRC$. 18 CA forks a child communication process, which asks U_d to send its DAK_NRC . 20 NO Received NO YES DAK_NRC from TIME-OUT? U_d ? YES Y Receive a snapshot copy of $CA_DAK[U_s]$ and $CA_DAK[U_d]$ and their counts CA $NRC[U_s]$ and CA $NRC[U_d]$ from their corresponding daemon. Then, CA aligns with U_s and U_d (FIG. 5) 24 Successful NO YES synchronization of both users? 26 CA ignores the last synchronization effects of the non-synchronized user, sends an "ABORT" message to both users, and terminates its child process. 28 CA authenticates both U_s and U_d (FIG. 6a) 30 Successful NO YES authentication of 34ء 32 both users? CA generates a dynamic session key DSK and sends a "SESSION_KEY" CA ignores the last synchronization message to U_s and U_d , including DSK encrypted by each user's dynamic effects of the non-authenticated authentication key (CA DAK $[U_s]$ and CA DAK $[U_d]$). The DSK along with user, sends an "ABORT" message the frozen/snapshot DAKs, at both user and CA nodes, are used as a new to both users, and terminates its state, in the DAK regeneration process, by the key management daemons. child process. Then, CA's child communication process terminates.

FIG. 4

Handy Solinan

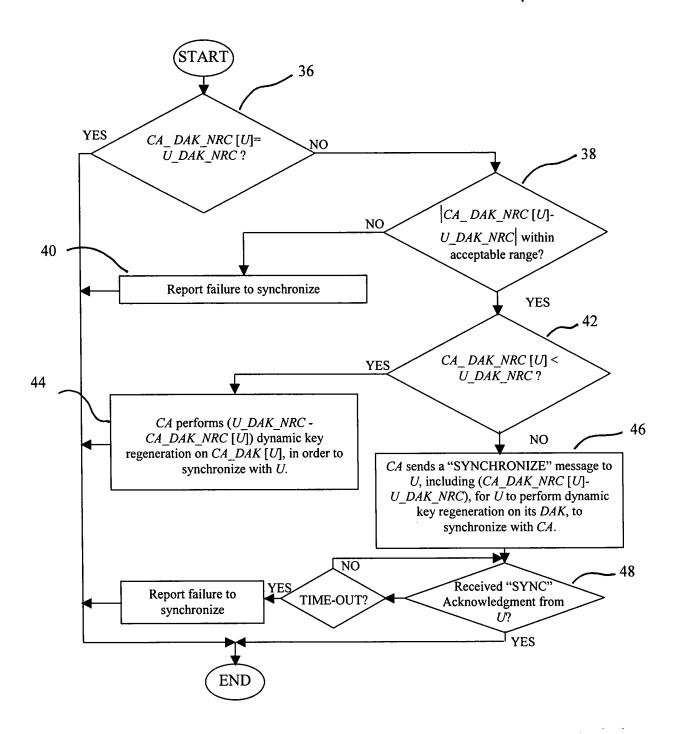


FIG. 5

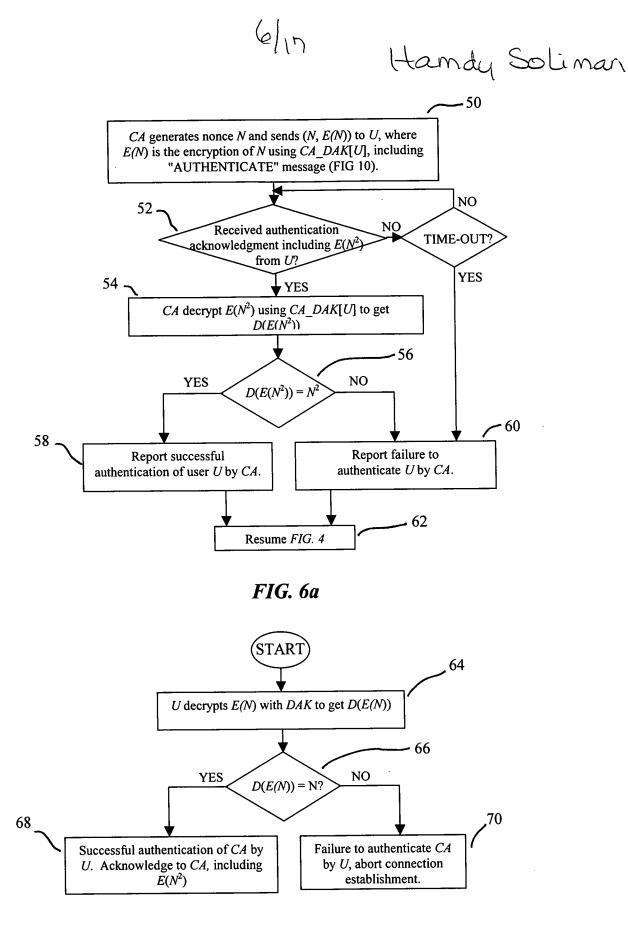


FIG. 6b

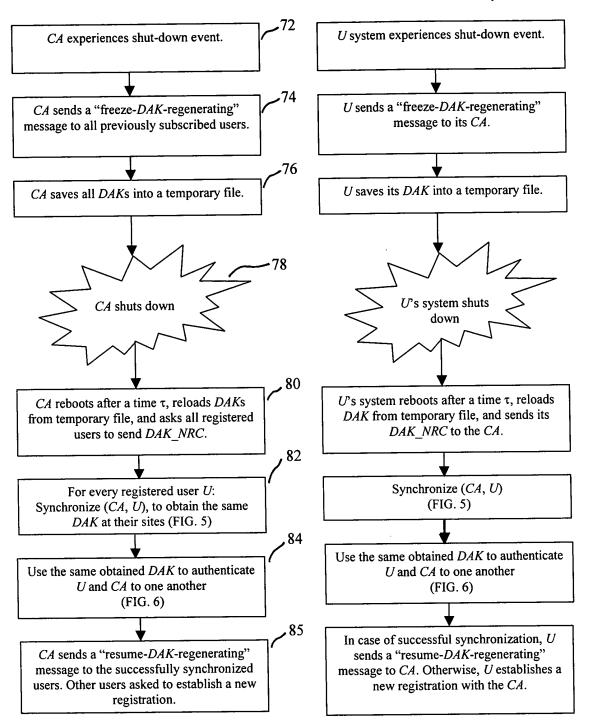


FIG. 7a

FIG. 7b

Handy Soliman

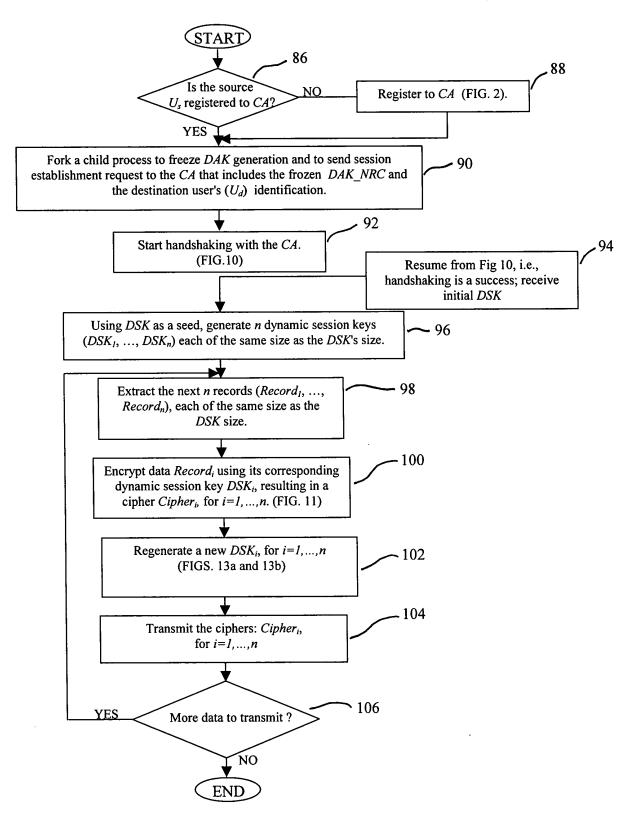


FIG. 8

Handy Soliman 108 U_d receives a request of communication with U_s from CA. -110 Fork a child process to stop regenerating DAK; send the frozen DAK_NRC to CA. 112 Start handshaking with the CA. (FIG.10) 114ر Resume from Fig 10, i.e., handshaking is a success; received initial DSK Using DSK as a seed, generate n new DSKs (DSK_1 , ..., -116 DSK_n) each of the same size as the DSK size. Receive the cipher records: Cipheri, -118 for i=1,...,n. 120 Decrypt cipher records Cipher, using corresponding DSK_i, resulting in a decrypted record $Record_i$, for i=1,...,n. (FIG. 12) 122 Restore the original message data by assembling decrypted records (Record₁, ..., Record,) -124 Regenerate new DSK_i , for i=1,...,n(FIGS. 13a and 13b) 126 YES More data to receive? NO. END

FIG. 9

Handy Soliman

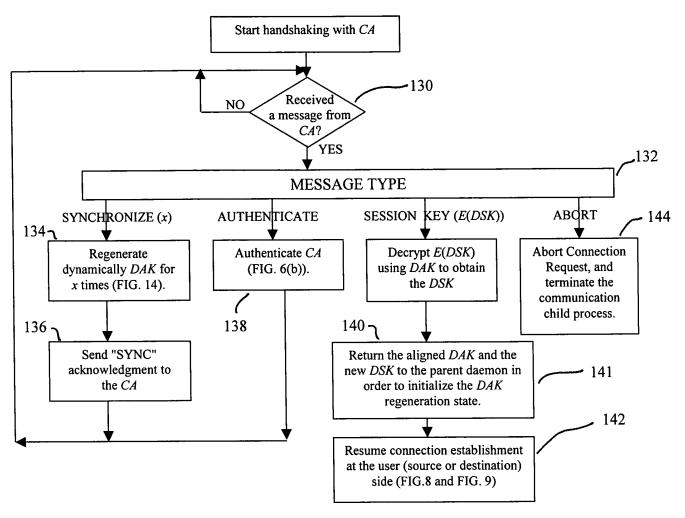


FIG. 10

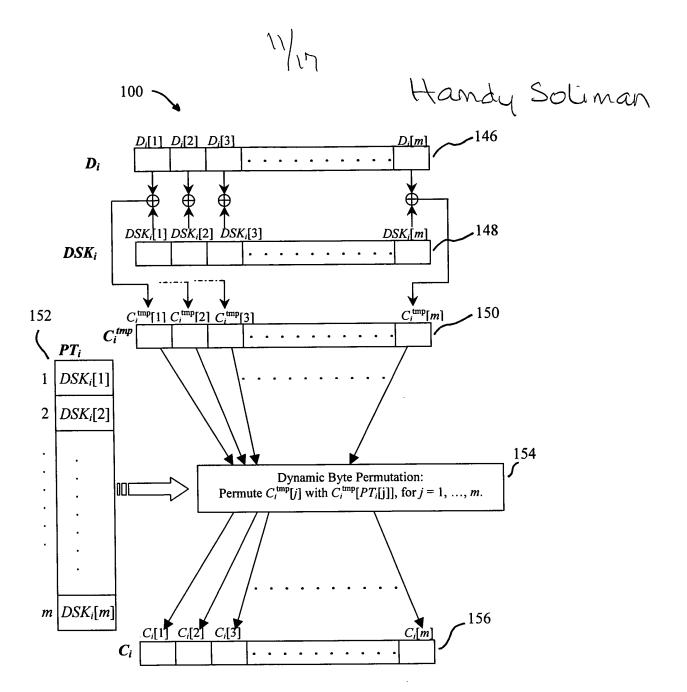


FIG. 11

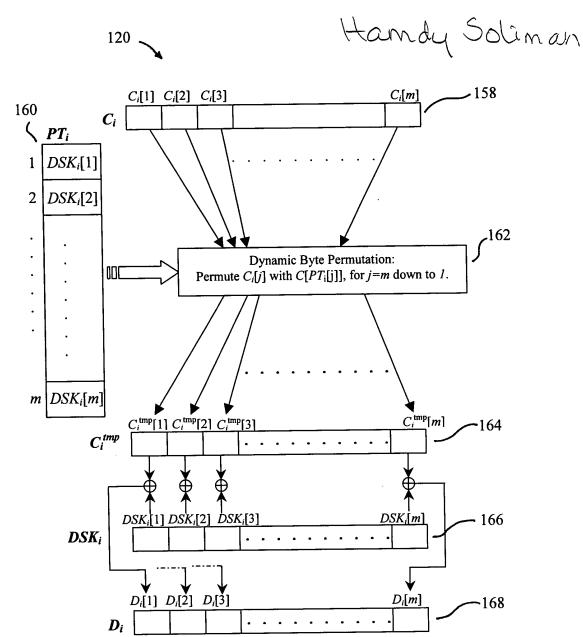
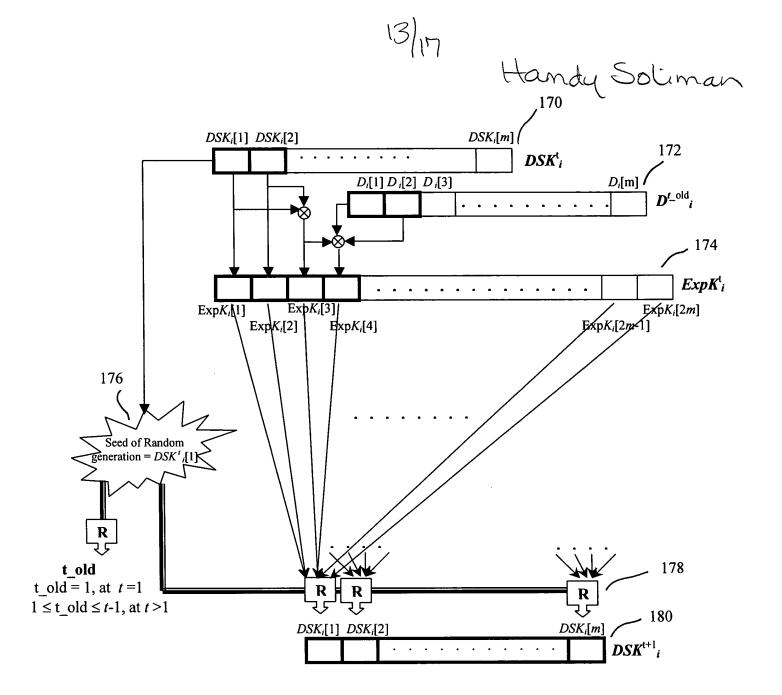
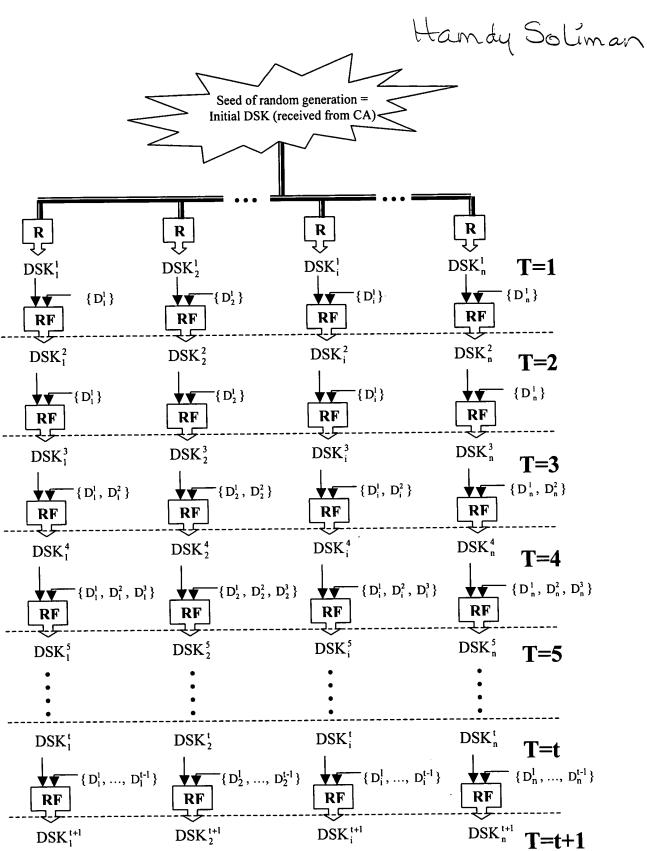


FIG. 12





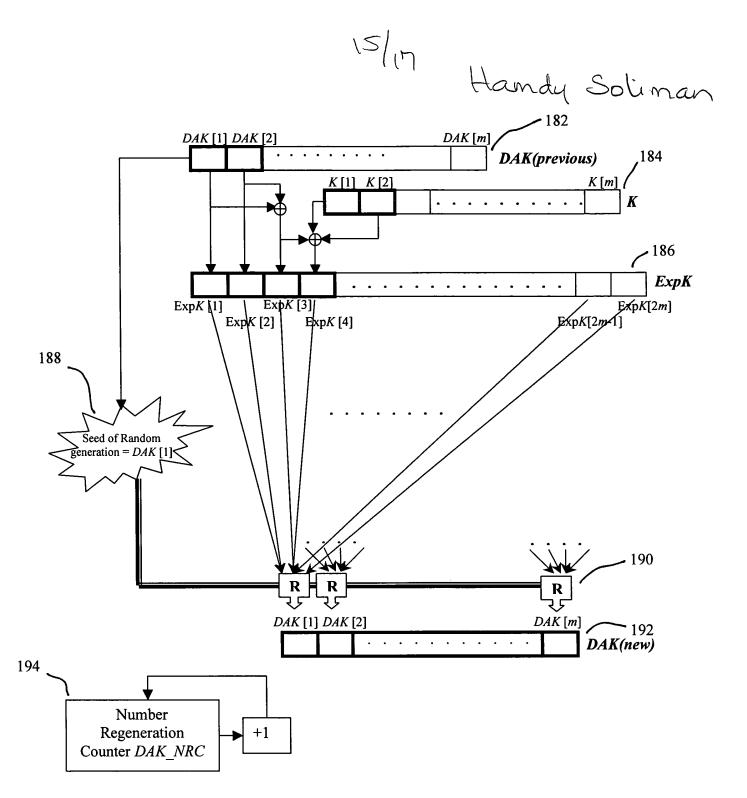


FIG. 14

16/17 Handy Solinan _196 DAK[m]DAK [1] DAK [2] 198 Initial DAK DAK[m]DAK [1] DAK [2] Initial DAK **ExpK** ExpK[2m]ExpK [3]/ ExpK[1]ExpK[4]Exp\([2] Seed of Random generation = DAK [1]< K[m]K[1]K[2]- 206

FIG. 15a

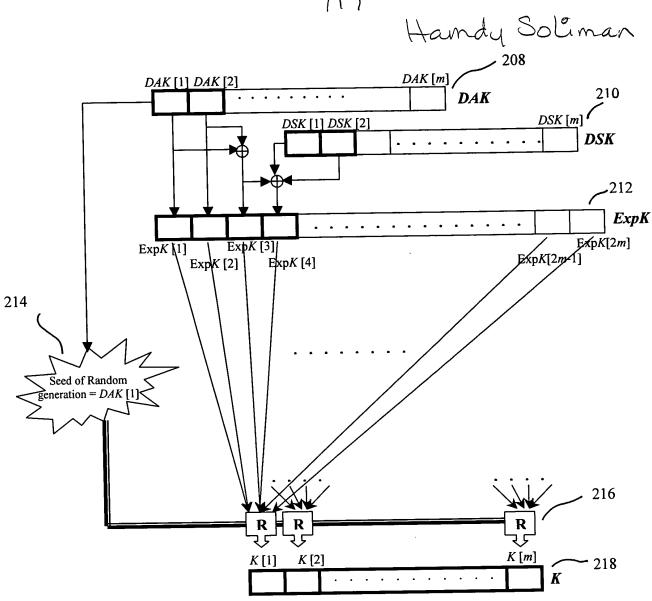


FIG. 15b